Lezione 4 Primi esempi di programmi

Vediamo un primo esempio di programma che utilizza l'istruzione div per effettuare la divisione intera fra una variabile a presente in memoria e 2, ponendo il quoziente della divisione in una variabile b. Per comprendere il programma ricordiamo che se il dividendo è a 16 bit, esso va posto nel registro ax ed il divisore dovrà essere a 8 bit e può essere una variabile di memoria o un registro ma non un valore espresso immediatamente nell'istruzione. Il quoziente sarà ad 8 bit e si troverà, al termine dell'istruzione, in AL.

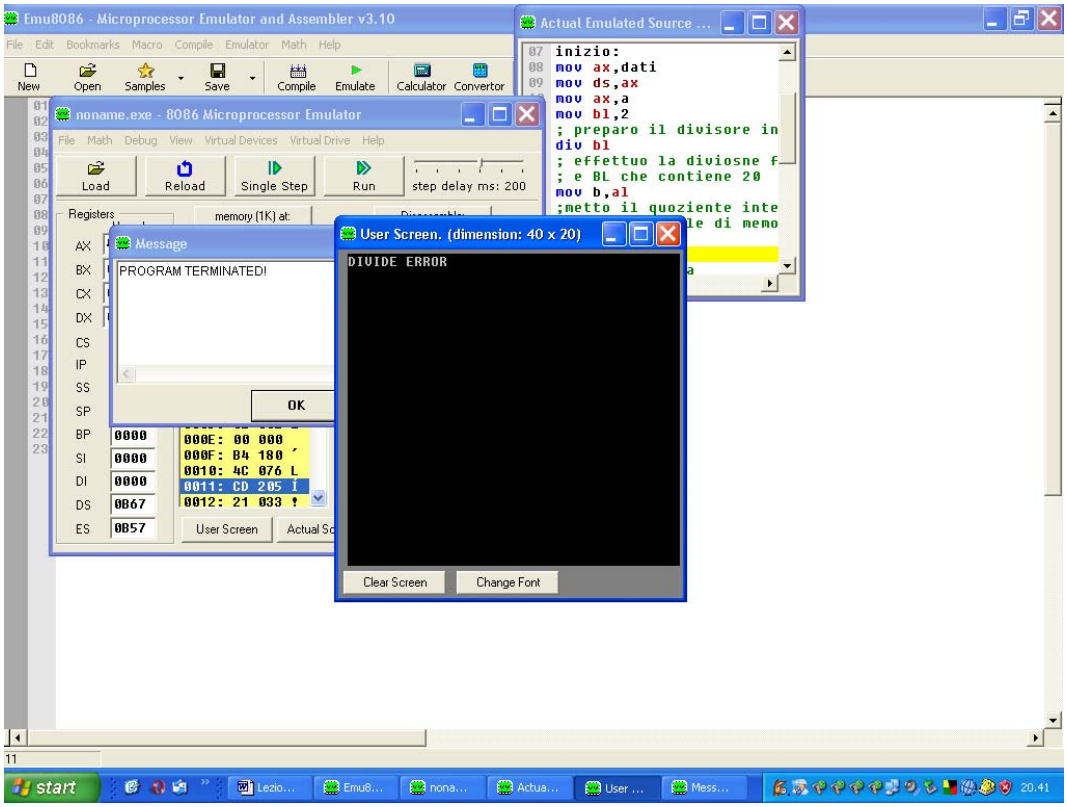
```
dati segment
a dw 1000
b db ?
dati ends
codice segment
assume cs:codice, ds:dati
inizio:
mov ax,dati
mov ds,ax
mov ax,a
mov bl,2
; preparo il divisore in BL
div bl
; effettuo la divisione fra AX che contiene la variabile a
; e BL che contiene 2
mov b,al
;metto il quoziente intero della divisione contenuto il AL
;nella variabile di memoria b
mov ah,4ch
int 21h
;fine programma
codice ends
```

end inizio

di seguito troviamo il link per i video della simulazione del programma.

- File avi
- File eseguibile

Come si può notare abbiamo un messaggio di errore



in realtà si è avuto un errore di overflow dovuto al fatto che, essendo il dividendo troppo piccolo, il risultato della divisione (500) non è esprimibile con gli 8 bit del registro AL. Con la nuova versione del programma non si ha questo problema

```
dati segment
a dw 1000
b db ?
dati ends
codice segment
```

```
assume cs:codice, ds:dati
inizio:
mov ax,dati
mov ds,ax
mov ax,a
mov bl,20
; preparo il divisore in BL
div bl
; effettuo la divisione fra AX che contiene la variabile a
; e BL che contiene 20
mov b,al
;metto il quoziente intero della divisione contenuto il AL
;nella variabile di memoria b
mov ah,4ch
int 21h
;fine programma
codice ends
end inizio
```

- File avi
- File eseguibile

Vediamo ora un esempio di moltiplicazione fra numeri interi senza segno utilizzando l'istruzione mul. In questo caso si tratta di un prodotto ad 8 bit per cui il primo operando si troverà in AL e il secondo operando dovrà essere ad 8 bit e risiedere o in una variabile di memoria o in un registro ad 8 bit. Il risultato si troverà in AX.

```
dati segment
a db 100
b dw ?
;la variabile b deve contenere il risultato della
```

```
;moltiplicazione
dati ends
codice segment
assume cs:codice, ds:dati
inizio:
mov ax,dati
mov ds,ax
mov al,a
mov bl,2
; preparo il moltiplicando in BL
mul bl
; effettuo la moltiplicazione fra AL che contiene la variabile a
; e BL che contiene 2
mov b,ax
;;il risultato della moltiplicazione è a 16 bit e si trova in AX
mov ah,4ch
int 21h
;fine programma
codice ends
end inizio
```

- File avi
- File eseguibile

Vediamo ora una moltiplicazione a 16 bit. Il primo operando va inserito nel registro ax e il secondo operando deve essere a 16 bit (variabile di memoria o registro). Il risultato sarà a 32 bit e la parte bassa sarà in AX e la parte alta in DX

```
dati segment
a dw 1000
blow dw ?
bhigh dw ?
;la variabile blow deve contenere la parte bassa del risultato della
;moltiplicazione
;la variabile bhigh deve contenere la parte alta del risultato della
;moltiplicazione
dati ends
codice segment
assume cs:codice, ds:dati
inizio:
mov ax,dati
mov ds,ax
mov ax,a
mov bx,100
; preparo il moltiplicando in BH
mul bx
; effettuo la moltiplicazione fra AX che contiene la variabile a
; e BX che contiene 2
mov blow,ax
mov bhigh,dx
mov ah,4ch
int 21h
;fine programma
codice ends
```

end inizio

- File avi
- File eseguibile